

10/726,917

PCTWORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 15/00		A1	(11) International Publication Number: WO 00/62178
			(43) International Publication Date: 19 October 2000 (19.10.00)
(21) International Application Number: PCT/US00/09646 (22) International Filing Date: 11 April 2000 (11.04.00) (30) Priority Data: 09/291,121 14 April 1999 (14.04.99) US (71) Applicant: VARIS CORPORATION [US/US]; 7500 Innovation Way, Mason, OH 45040 (US). (72) Inventors: GAUTHIER, Forrest, P.; 8792 Maineville Road, Maineville, OH 45039 (US); WALKER, James, R.; 3151 Madera Lane, Maineville, OH 45039 (US). (74) Agents: LEVY, Mark, P. et al.; Thompson Hine & Flory LLP, 2000 Courthouse Plaza N.E., P.O. Box 8801, Dayton, OH 45401-8801 (US).		(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG). Published With international search report.	
(54) Title: METHOD AND SYSTEM FOR MERGING VARIABLE TEXT AND IMAGES INTO BITMAPS DEFINED BY A PAGE DESCRIPTION LANGUAGE			
(57) Abstract A computer implemented method includes the steps of: a) generating a template PDL (14) specification defining how the template data is to appear on a printed page; b) generating a plurality of sub-templates PDL (15) specifications; c) interpreting the template specification so as to generate bitmap or commands (display list); d) saving the bitmap or commands into memory; e) associating the variable data identifier with the sub-template specifications; f) accessing a first sub-template specification; g) processing the first sub-template specification to generate bitmap or commands; h) accessing the bitmap or commands; and i) merging plurality of rendering commands (16), if necessary. The steps may be repeated, as necessary, to generate a plurality of merged bitmaps.			

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece	ML	Mali	TR	Turkey
BG	Bulgaria	HU	Hungary	MN	Mongolia	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MR	Mauritania	UA	Ukraine
BR	Brazil	IL	Israel	MW	Malawi	UG	Uganda
BY	Belarus	IS	Iceland	MX	Mexico	US	United States of America
CA	Canada	IT	Italy	NE	Niger	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NL	Netherlands	VN	Viet Nam
CG	Congo	KE	Kenya	NO	Norway	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NZ	New Zealand	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	PL	Poland		
CM	Cameroon	KR	Republic of Korea	PT	Portugal		
CN	China	KZ	Kazakhstan	RO	Romania		
CU	Cuba	LC	Saint Lucia	RU	Russian Federation		
CZ	Czech Republic	LI	Liechtenstein	SD	Sudan		
DE	Germany	LK	Sri Lanka	SE	Sweden		
DK	Denmark	LR	Liberia	SG	Singapore		
EE	Estonia						

**METHOD AND SYSTEM FOR MERGING VARIABLE TEXT
AND IMAGES INTO BITMAPS DEFINED BY A PAGE DESCRIPTION
LANGUAGE**

CROSS-REFERENCE TO RELATED APPLICATIONS

The present patent application is a Continuation-In-Part of U.S. Patent Application, S/N 08/896,899, filed January 18, 1997; which is a Continuation-In-Part of U.S. Patent Application, S/N 08/373,582, filed January 18, 1995 and issued as U.S. Patent No. 5,729,665.

BACKGROUND OF THE INVENTION

The present invention relates to the high-speed printing industry, and more particularly, to a system and method for merging variable data and images into a template image defined by a page description language file in a high speed printing environment.

Application programs, such as wordprocessors, illustrators, and computer aided design systems are software packages used to create a document (text and graphics) on a computer screen and to simultaneously generate a page description language ("PDL") specification, which is to be transferred to the printer, or to any other type of raster device or output device for creating a hard copy or copies of the document. Alternatively a PDL specification can be generated by a programmer without the assistance of an application program.

The printer executes the PDL specification to generate a bitmap of the document, or a raster-data representation of a document, and eventually transfers the bitmap or raster-data to a physical medium such as paper. A typical PDL language, such as PostScript (a registered trademark of Adobe Corporation) defines a page of the document as containing a number of data areas, where each data area contains either graphic or alpha-numeric data. Each data area is defined by a "graphic state," which is a collection of parameters or attributes for controlling the representation and appearance of text and graphics. For example, the graphic state can include a set of text attributes such as scale factor, type font, etc. In postscript an example of a PDL command used to build a graphic state can be:

20 rotate /Times-Roman findfont 14 scalefont setfont

Examples of PDL commands used to define the graphic or alpha-numeric data that is displayed in the data area include:

0 0 moveto and (ABC) show

The entire group of PDL commands used to define a document is hereinafter referred to as the "PDL specification." Furthermore, the entire graphic state, or any particular attribute or combination of attributes included in a graphic state, or any similar attribute contained in a PDL specification for defining or controlling the representation, location and/or appearance of text and graphics in a final bitmap or raster image is hereinafter referred to as "graphic attributes."

In variable data printing each printed document shares a common template and there is at least one area in the template that changes for each printing of the template. Typical PDL languages are not designed for high-speed variable data printing because, with PDL languages and interpreters, even if a single item of data in the document changes, an entirely new PDL specification must be created and interpreted. For example, if 100,000 copies of a mass mailing advertisement were to be printed (i.e., each copy of which is identical except for the mailing address) it is typically necessary to generate a new PDL specification for each copy to be printed. Hence, to generate 100,000 advertisements, it would be necessary to generate 100,000 PDL specifications, even though each advertisement is virtually the same except for the variable data area. The processing time required to interpret and render 100,000 PDL specifications is enormous, significantly slowing the entire printing system.

Furthermore, typical PDL languages do not include the capability of rapidly merging variable images or bitmaps (such as company logos, coupons, charts, and the like) along with variable text data into the template bitmaps. Accordingly, there is a need for a high-speed printing operation having the ability to merge variable data (which includes variable text data and bitmap images) into a template defined by a PDL specification.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a system and method for merging variable text and bitmap images into a PDL specification in high-speed printing operation. It is a further object of the present invention to provide the ability to generate a plurality of merged bitmaps, which are each essentially a copy of the template, except for at least one portion of the template into which the variable data has been merged. In this portion, each merged bitmap can contain a different set of variable data merged into it. The template is defined by a PDL specification, and this template specification only needs to be processed or interpreted once before creating all of the merged bitmaps, thus providing an extremely high-speed variable data printing operation. The variable images to be merged into the template may also be defined by "sub-template" PDL specifications, which also need only be processed or interpreted once. Such sub-template specifications may also allow for variable data or images to be merged into them as well, before being merged into the primary template.

A computer implemented method for merging variable data into an image defined by page description language specification ("PDL specification"), according to the present invention, generally comprises the steps of: processing (interpreting) the template PDL specification to produce a template; processing the sub-template specification (defining the variable bitmap image) to produce a sub-template; identifying a variable data identifier in the template specification; associating the sub-template with the variable data identifier; and merging the sub-template into a copy of a template to generate a merged bitmap.

More specifically, a computer implemented method of the present invention comprises the steps of: a) generating a template PDL specification, the template specification including template data and associated graphic attributes (i.e., graphic states) defining how the template data is to appear on a printed page, the template specification including at least one variable data identifier; b) generating a plurality of sub-template PDL specifications, each sub-template specification including sub-template data and associated graphic attributes defining how the sub-template data is to appear on a portion of a printed page; c) interpreting the template specification so as to generate a template bitmap or a plurality of template rendering commands (display list), and during the interpreting

step, identifying the variable data identifier; d) saving the template bitmap or the plurality of template rendering commands into memory; e) associating the variable data identifier with the sub-template specifications; f) accessing a first sub-template specification from the plurality of sub-template specifications; g) processing the first sub-template specification so as to generate a sub-template bitmap or a plurality of first sub-template rendering commands; h) accessing a copy of the template bitmap or the plurality of template rendering commands from memory; i) merging the copy of the template bitmap or template rendering commands with the sub-template bitmap or sub-template rendering commands so as to provide a first merged bitmap or first merged plurality of rendering commands; j) generating a first merged bitmap from the first merged plurality rendering commands (if necessary); k) accessing a next sub-template specification from the plurality of sub-template specifications; l) processing the next sub-template specification so as to generate a next sub-template bitmap or plurality of next sub-template rendering commands, m) accessing a copy of the template bitmap or template rendering commands from memory; n) merging the copy of the template bitmap or template rendering commands with the next sub-template bitmap or sub-template rendering commands so as to provide a next merged bitmap or next plurality of rendering commands; and o) generating a next merge bitmap from the next merged plurality of rendering commands, if necessary. The steps k-o may be repeated, as necessary, to generate a plurality of merged bitmaps.

The method of the present invention is accomplished by executing a control task in conjunction with a PDL interpreter program. The control task generates a template display list based upon the PDL commands in the PDL specification. The display list includes a plurality of rendering commands, where each rendering command designates a particular data area or object to be rendered, the graphic states to be applied to the data area and the offset address at which the rendered object, if any, in the data area is to be over written onto the final bitmap. The graphic states for each data area re set forth in the PDL specification, and pertain to the print attributes that describe how particular graphic or alpha-numeric data is to appear on the printed page. These attributes can include size, font, position, orientation, location and the like. The control task also generates display lists for each of the sub-template PDL specifications.

The control task, during the PDL interpretation procedures, monitors the data areas defined by the PDL specifications to watch for variable data identifiers defined by the PDL code. If the control task identifies a data area or object as being (or including) a variable data identifier, it reserves the graphic states associated with that variable data identifier in a cache and continues on with the interpretation procedure, preferably without adding the rendering commands for that variable data area into the display list. In this identification step, the control task will also watch for attributes associated with the variable data identifier. Such attributes may define the variable data identifier as identifying a sub-template bitmap, which is to be merged into the bitmap represented by the PDL specification. If the interpreter detects such an attribute, rather than saving off the graphic states associated with this variable data area, the control task will instead store a "place holder" in the graphic states cache corresponding to the sub-template PDL specification identified in the variable data identifier string. In certain embodiments of the invention, the control task may include certain graphic attributes associated with the variable data identifier in the cache, along with the place holder, such as a graphic attribute defining the location of the variable data identifier (which may be used to determine where to merge the sub-template bitmap into the template bitmap).

Once the interpreter program completes its interpretation of the PDL specifications, the control task saves each display list in memory without dispatching a bitmap of the template or sub-template to the printer. Subsequently, the merge task is initiated which accesses a copy of the template display list from memory and begins processing the rendering commands in the display list to create a bitmap of the template. Also, the merge task accesses a variable data record from a merge file; associates the variable data record to the graphic states in the cache and creates bitmaps for the data in the variable data record by applying the reserved graphics states to the data in the variable data record these bitmaps may be merged into the template bitmap or rendered directly onto the template bitmap during, before, or after the rendering of the template bitmap. When the merge task reaches the place holder in the cache associated with the sub-template display list, the merge task will access and begin processing the rendering commands in the sub-template display list to create a bitmap of the sub-template, which is then merged onto the template bitmap. When finished with the rendering commands of the sub-template display

list, the merge task will return to the processing of the variable data record by applying the reserved graphic states thereto and merging the resulting bitmaps into the template bitmap. The merge task is repeated for each variable data record in the merge file to create a plurality of the merged bitmaps.

Therefore, the PDL specifications of the template and sub-templates need only be interpreted once, saving significant processing time for the variable printing operation, because the reserved graphic states may be utilized over and over again to create the variable bitmaps for each variable data record contained in the merge file. Similarly the display lists of the templates and sub-templates may be used over and over again to create the multiple merged bitmaps.

In its simplest form a sub-template is a collection of graphic states or graphic attributes taken from another source (such as a PDL specification) and used collectively as a variable element inserted into a template PDL page. The sub-templates may be used to add variable graphics/logos, or other static images to a page; to add formatted pieces (such as graphs, charts, images, etc.), including any number of additional variable fields to a page, a page of variable coupons directly marketed to the end receiver, and the like. The sub-template is really just another PDL page used as an element drawn somewhere on another page. For example, one could have a PDL page that had a company logo drawn on it. One could then use that logo on demand whenever a printed page calls for it. Taken further, one could have several different logos, each drawn as a separate PDL page, which one could add (or not) to any page printed during the processing job.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a schematic, block diagram representation of a high-speed printing system according to the present invention;

Fig. 2 is a schematic flow diagram illustrating the method of the control task according to the present invention;

Fig. 3 is a schematic flow diagram representing the method of the merge task according to the present invention;

Fig. 4 is an example of a job ticket file for use with the present invention;

Fig. 5 is an example of a merge file for use with the present invention;
Fig. 6 is an example of a sub-template file for use with the present invention.
Fig. 7 is an example of a template file for use with the present invention; and
Fig. 8 is an example of a merged page created by the process of the present invention using the PDL specification of Fig. 7, the job ticket of Fig. 4, the sub-template specifications of Fig. 6 and the merge file of Fig. 5.

DETAILED DESCRIPTION

As shown in Fig. 1, a system for performing the method of the present invention includes a printer controller 10 having access to a job ticket file 12 a page description language ("PDL") file of a template 14, a plurality of PDL files for sub-templates 15, a source of variable data such as a merge file 16 and an optional printer configuration file 18. The system also contains an operator control terminal 20 for providing operator controls such as indicating the name and path of the job ticket file 12 for the specific print job.

The job ticket file 12 contains the guidelines for the print job which can include the names and locations of the PDL files 14, 15, the merge files 16, the configuration files 18, etc.; and may also include special instructions pertaining to the print job such as identifying and locating sub-templates, defining additional graphical attributes for variable data areas identified during the process, and the like, all of which is described in greater detail below. The PDL files 14, 15 are preferably PostScript® specifications created by an application program such as a wordprocessor, illustrator, or computer aided design system. The merge file 16 contains platform independent data, such as text data, image data, bar-code data and the like, which is to be merged into a template bitmap defined by the PDL template specification 14 or the PDL sub-template specifications 15 during the merging task, as will be described in detail below. The configuration file 18, defines the print engines and the post processing equipment and other options to be executed.

Initially, the path and name of the job ticket file 12 is specified by the operator using the operator control terminal 20. The printer controller 10 retrieves the job ticket file 12 and then retrieves the PDL files 14, 15 specified in the job ticket file. Next,

the controller 10 initiates a control task 22 in conjunction with a page description language interpreter program.

The control task 22 interprets the PDL specifications from the PDL files 14, 15 and monitors data areas defined in the PDL specifications to watch areas defined by the specifications to become variable. If the control task identifies a data area as being a variable data area, it reserves the graphic states and/or other associated graphic attributes 23 of that variable data area in a cache or memory 24 and then moves on to the next data area defined by the particular PDL specification, usually without allowing any data defined by the variable data area to be added to the template bitmap. The control task also looks for predetermined attributes defined in the data areas to determine if the data area is defining the importation of a sub-template bitmap. If the control task detects such an attribute, rather than storing the graphic states associated with the data area in the cache 24, it stores a placeholder 23' in memory, which will instruct the merge task that a sub-template is to be incorporated into the present template or sub-template bitmap during the merge task 28, and will also include information identifying the one or a group of the sub-templates that may be merged into the present template or sub-template bitmap. Once the control task completes its processing of the particular PDL specification, the control task saves the template display list 25 or sub-template display list 27 in memory 26. The template and sub-template display lists 25, 27 will include a plurality of rendering commands for the static data defined in their respective PDL specifications. Each rendering command designates a particular static data area or object to be rendered, the graphic states and/or graphic attributes to be applied to the static data area and the offset address at which the rendered object, if any, in the static data area is to be over written onto the final bitmap.

Next, a merge task 28, having access to the variable data records 17 from the merge file 16 is executed to apply the reserved graphic states and/or graphic attributes 23 to the variable data records 17, creating rendering commands for that variable data record as defined by the graphic states. The merge task 28 retrieves a copy 25' of the template display list from the memory 26 and merges the variable data rendering commands with the template display list to create a merged display list 30. The merge task will also look for place holders 23' among the graphic states stored in the memory 24 during this merging

operation. If a place holder 23' is detected, the merge task will access a copy 27' of the display list of the sub-template corresponding to the place holder and will then merge the rendering commands from the display list of the sub-template 27' with the merged display list 30. It is noted that the sub-template may also include an associated cache of graphic states and/or graphic attributes corresponding to variable data areas (or even additional levels of sub-templates) defined within the sub-template. Therefore, if such a cache is present with a particular sub-template, the merge task will apply such stored graphic states and/or graphic attributes to the present variable data fields in the variable data record 17 linked to the graphic states to therefore create rendering commands for such variable data fields. These rendering commands are also merged into the display list 30.

Once the merged display list 30 is created, the controller 10 performs a rendering task 32 to render the merged display list 30 into a plurality of bitmap bands 34 for dispatching to at least one print engine 36. A general method for performing the above control task is described in U.S. Patent No. 5,729,665, the disclosure which is incorporated herein by reference. A method and system architecture for performing the above merging, banding and dispatching operations are respectively described in U.S. Patent Nos. 5,594,860 and 5,796,930, the disclosures of which are also incorporated herein by reference.

As shown in Fig. 2, a graphical flow diagram representation of the control task is illustrated. As discussed above, the primary function of the control task is to monitor a PDL interpreter program which interprets the PDL specifications for the template (14) and the sub-templates 15 to create display lists 25, 27, containing the rendering commands for the static data in the PDL specifications, and a cache of graphic states and/or graphic attributes corresponding to the variable data areas identified by the PDL specifications. While the PDL specifications are typically in the form of a list of PDL commands (as described above) the specifications 14, 15a and 15b shown in Fig. 2 are shown, for clarity, as they would have appeared to the artist using the application program (such as QuarkXPress[®]) to create such PDL specifications.

As shown in Fig. 2 the template PDL file 14 includes a plurality of static data areas 38 and a plurality of variable data areas 40. The variable data areas are identified by the control task as a text string surrounded by special characters, "<<" and

"> >". The phrase or word within the special characters corresponds to the field name for the particular variable data area. These strings may also be followed by an attribute command string 42, which may define special attributes to be applied to the particular variable data area. With respect to the template PDL file shown in Fig. 2, the variable data identifier with the field name "PICTURE" is followed by the attribute command string "SELECT", which, as will be discussed below, informs the control task that the particular variable data area corresponds to the insertion of a sub-template.

As discussed above, the rendering commands for the static data areas 38 in the template PDL file are stored in a template display list 25 and the graphic states 23 of the variable data areas 40 are stored in a cache 24. As also discussed above, and as will be discussed in detail below, the graphic state stored in the cache for the variable data area having the "select" attribute string is merely a place holder 23', which will instruct the merge file to insert a sub-template bit map into the template bit map being rendered.

The control task will also interpret the sub-template PDL files 15a, 15b, shown as sub-template "A" and sub-template "B" in Fig. 2. Sub-template "A" 15a includes two static data areas 38, and therefore a sub-template display list 27a is created for the sub-template "A" PDL file, including the rendering commands for such static data areas. The sub-template "B" PDL file 15b includes a static data area 38 and a variable data area 40. Therefore, the control task will create a sub-template display list 27b, including the rendering command for the static data area and will cache the graphic states 23' for the variable data area. It is noted here that the variable data identifiers in the sub-template PDL specifications may also include attribute strings for special processing, such as specifying additional levels of sub-template files. For the purposes of simplicity, the present example is shown with only one level of sub-templates.

As shown in Fig. 3, the merge task will access a copy 25' of the template display list and will also access a first record 46 from the merge file 16. Note that the first record has record fields, in the form of text strings, for each of the field names VAR1, VAR2, VAR3 and VAR4. For the field name PICTURE, the record includes a name (in this example either A or B) corresponding to the sub-template to be inserted for the variable data identifier 40 having the field name "PICTURE." The merge task accesses the rendering commands from the copy of the template display list 25' to add to a merged

display list 30. Such rendering commands will be processed by the rendering task 32 to generate the bit maps 48 for the static data areas from the template PDL specification 14 to appear in the merged bitmap 52. The merge task will link the cached graphic states 23 to the to the record fields by matching the field names associated with the cached graphic state to the field names in the merge file. For example, the cached graphic states for the variable data identifier 40 having the field name "VAR1" will be linked to the record fields in the merge file under the field name "VAR1." Once linked, the merge task will apply the graphic states 23 to the data in the associated record field to create rendering commands for such data, which are to merged into the merged display list 30. Such rendering commands will be processed by the rendering task 32 to generate the bit maps 50 in the merged bitmap 52. This is done for each of the graphic states in the cache.

When the merge task reaches the place holder 23' in the template graphic states cache, it will link the place holder 23' to the to the record fields by matching the field name associated with the the place holder 23' to a field name in the merge file. For example, the he place holder 23' for the variable data identifier 40 having the field name "PICTURE" will be linked to the record fields in the merge file under the field name "PICTURE." The field name in the record will identify which of the sub-templates to merge into the merged display list 30 and eventually into the merged bitmap 52. Because the merge record 46 indicates that the sub-template "B" is to be used, the merge task accesses the sub-template display list 44b and merges its rendering commands into the merged display list 30. Such rendering commands will be processed by the rendering task 32 to generate the bit maps 54 in the merged bitmap 52. The merge task also accesses the associated cache of graphic states for the sub-template, and applies the graphic states to the record fields linked to the graphic states so as to generate rendering commands for the data of the record fields that are to be merged into the merged display list 30. Such rendering commands will be processed by the rendering task 32 to generate the bit maps 56 in the merged bitmap 52.

Note that if this sub-template "B" included a further level of sub-templates, the cache graphic states for the sub-template would also include a place holder, and the merge task would access the sub-templates associated with this place holder for another level of sub-template processing.

The final merged big map 52 includes the static data bitmaps 48 defined by the template PDL file 14, the static data bitmaps 54 defined by the sub-template "B" PDL file 15b, the variable data bitmaps 50 having the graphic attributes corresponding to the cached graphic states for the variable data identifiers 40 in the template PDL file 50, and the variable data bitmaps 56 having the graphic attributes corresponding to the cached graphic states for the variable data identifiers 40 in the sub-template "B" PDL file 15b. The location of the bit maps 54, 56 from the sub-template "B" PDL specification can be defined by the job ticket file (see the Appendix to this disclosure). Furthermore, it is within the scope of the invention to include a graphic state or graphic attribute with the place holder that corresponds to the location of the variable data identifier (this graphic attribute may also include other information such as orientation, size, etc.). This additional graphic attribute may be stored with the place holder and applied to the bit map data from the sub-template file during the merging operation. For example, a graphic state corresponding to the location of the variable data identifier having the attribute 42 corresponding to the sub-template may be stored with the place holder in the graphic state cache to direct the merge task to place the bit maps from the sub-template in the merge bit map 52 in the location directed by the stored graphic state.

An embodiment of the present invention is illustrated by way of example in Figs. 4-8. As illustrated in Fig. 4, the job ticket file 12 contains a group header 60 "[PageDescriptionLanguageFile]" indicating that the phrases below that group header 60 are the names of page description language files to be processed by the control task. In the present example, there are two page description language files: a file "letter_master" 62 and a file "all_maps" 64. The job ticket file 12 includes a group header 66 "[letter_master]," under which are defined the location and attributes for that PDL file. Note that with this PDL file no attributes are defined and only the path location for the file is indicated. Another group header 68 "[all_maps]" defines the path and attributes for the all_maps PDL file. Note that an attribute string 70 "SubTemplate = true" indicates that this file is a sub-template file and the attribute string 72 "Templates = South, East, West, Midwest" indicates that the file includes four sub-templates, named South, East, West and Midwest.

The job ticket also includes a group header 74 "[MergeFiles]" identifying the names of the merge files to be used in the merge task. In the present case a single merge

file 76 is named "merge." Below that, a group header 78 "[merge]" is given, under which the attributes and location of the merge file is set forth. The attribute strings for this merge file indicate that the merge file is delimited and includes merge headers. The attribute strings also indicate that the records are delimited by a carriage-return/line-feed character and the particular fields in each record are delimited by a tab character. A complete description of the different attributes that can be defined for the PDL files is described in detail in the Appendix below.

As illustrated in Fig. 5, the merge file 16 has a platform-independent data file that contains the "variable" data to be merged into the path defined in the PDL specification, and also includes names associated with the sub-templates to be merged into the PDL specification during the merge task. The merge file in the present example includes a plurality of rows of merge records separated by carriage-return/line space characters, where each record includes the following fields: "fname," "lname," "prefix," "title," "company," "address," "zip," "region," "city" and "state." As will be described below, the data in the field "region" doubles as variable data and also as a name of a sub-template to be merged into the final bit maps.

As illustrated in Fig. 6, the designer will utilize an application program to create a document containing static data and variable data identifiers. The application program will then be directed to create a PDL specification of the document by the designer. The variable data identifiers 80 each include a field name surrounded by special characters, "<<" and ">>", and may also include an attribute string 82 following the field name and special characters. The attribute string will be described below. The PDL specification generated by the application program will include the graphic states of the variable data identifiers 80. These graphic states can include the font size (i.e., 12 point), the type-font (i.e., Helvetica), the orientation (i.e., horizontal), the location such as x and y coordinates, and the like. As discussed above, the control task will create a display list with the rendering commands for the static data in the PDL specification and will cache the graphic states for the variable data identifiers without transferring the rendering commands for the variable data identifiers to the display list. The variable data identifier in the bottom center portion of the page includes the "select" attribute, indicating to the control task that a sub-template bit map is to be inserted into the document. The field name "region"

associated with this variable data identifier indicates that the sub-template name will be under the heading "region" in the merge file.

Referring to Fig. 5, the regions named in the merge file are South, East, West and Midwest. Therefore, referring to Fig. 4, the sub-templates having the name South, East, West and Midwest are found in the PDL file defined under the "all_maps" group header (68) in the job ticket. Note that the attributes defined under the "all_maps" group header do not include an attribute directing the location of the sub-template. Therefore, the lower left-hand corner of the sub-template merged into the final document will be directed by the locational graphic states of the variable data identifier stored along with the place holder in the graphic state cache.

Fig. 7 illustrates the sub-template PDL file 15, including the four sub-templates: South 84a, East 84b, West 84c and Midwest 84d. The merge task will access a variable data record 86 from the merge file 16 and a copy of the display list for the template PDL file 25. As discussed above, the merge task will generate rendering commands for the variable data records in the merge file by applying the cached graphic states linked to the variable data records. These rendering commands will be merged into the merged display list along with the rendering commands from the display list for the template PDL file. The merge task will also merge the rendering commands from the display list of the sub-template named by the variable data records into the merged display list. The rendering task will process the rendering commands in the merged display list to generate the final merged bit map for the present variable data record 86. This bit map appears in Fig. 8.

Referring to Figs. 5, 6 and 8, the merge task will apply the cached graphic states linked to the variable data fields in the variable data record 86 as follows. The first graphic state in the cache will be for the first variable data identifier 80a. The field name for this variable data identifier 80a is "fname," which in the particular variable data record, corresponds to "Shannon." Therefore, the final merged bit map will include a bit map of the text, "Shannon" 88a, having the graphic states for the variable data identifier 80a. Likewise, the second cached graphic state will be for the variable data identifier 80b which includes a field name "lname," corresponding to the term "Janzen" in the particular variable data record 86. Therefore, the final merged bit map will include a bit map of the text, "Janzen" 88b, having the graphic states for the variable data identifier 80b.

Correspondingly, the remaining graphic states, except for those of variable data identifier 88m, which is the variable data identifier including the sub-template attribute 82, will be processed in the same way resulting in bit maps 88c-88n in the merged bitmap.

When the merge task reaches the place holder in the graphic state cache associated with the variable data identifier 80m, it will refer to the merge record for the name of the sub-template under the field name "Region," which in the present example is "South." Referring to Fig. 7, the sub-template having the name "South" is sub-template 84a. Accordingly, the merge file will then merge the rendering commands from the display list of the sub-template 84a into the merged display list. Some or all of these rendering commands will also be modified by a graphic attribute corresponding to the location that the sub-template is to be merged into the merged bit map. As discussed above, this graphic attribute may be taken from the graphic state of the variable data identifier 80m or may be defined in the job ticket file.

The present invention also provides for the flowing of sub-templates into a path defined by the template specification. Such a feature is based upon the invention disclosed in U.S. Pat. App. Ser. No. 08/897,467, filed July 18, 1997. Specifically, the feature includes the steps of: associating a path defined by the template PDL specification with the variable data identifier with the "sub-template" attribute string; and merging the sub-template(s) into the path according to the path boundary and according to a predefined flow rule (as will be defined in the job ticket). The path may be associated with the attribute string, for example, by having the PDL command for the path immediately follow the PDL command for the attribute string in the PDL specification, or by having the PDL command for the path "grouped" with the PDL command for the attribute string using a GROUP command provided by the application program.

The following Appendix provides a preferred compilation of commands and parameter definitions that can be specified in the job ticket file 12 for the sub-template application as described above. Each entry provides a particular command header, the syntax for the command, any relevant remarks for the use of the command, examples, etc. As will be appreciated by one of ordinary skill in the art, the present invention includes any and all additional functions, features and attributes detailed in the Appendix.

APPENDIX**[PageDescriptionLanguageFile]**

A group that provides a list of tags which you create to describe the PDL file(s) to be used in the print job. Each tag will become a user-defined group to give additional information about a specific PDL file.

Syntax**[PageDescriptionLanguageFile]***PDL File Tag A**PDL File Tag B***Remarks****Required**

The number of tags listed equals the number of PDL files that VariScript is to interpret during the job. Every tag that appears under this initial [PageDescriptionLanguageFile] group will become a new group name in succeeding sections of the Job Ticket.

Explanation*PDL File Tag A*

Create a descriptive name for the first PDL file used in the print job. This tag is for use within the Job Ticket only and is not used outside of that context.

PDL File Tag B

Create a descriptive name for the next PDL file used in the print job. This tag is for use within the Job Ticket only and is not used outside of that context.

Example [PageDescriptionLanguageFile]

Cover Form

Contents Form

Letter Form

A user-defined tag name for a group that provides information about a PDL file and corresponds to a descriptive tag that you created under the initial [PageDescriptionLanguageFile] group.

Syntax [*PDL File Tag*]

File Path =

<other host access parameters>

<VariScript rendering parameters>

Templates =

Sub Template =

Sub Template Area =

See Also Templates

Remarks A separate [*PDL File Tag*] group is required for each descriptive tag listed under the initial [PageDescriptionLanguageFile] group.

Case sensitivity of the values that you define depends on the host operating system.

Explanation [*PDL File Tag*]

Take the descriptive tag under the initial [PageDescriptionLanguageFile] group and write it here as a group name within brackets [].

File Path =

Write the drive, path, and file name for the PDL file being described. The format of your notation is dependent on your host computer. See the FilePath element description in Chapter 3.

<other host access parameters>

Define values for any other host access parameter for which the default value is not accurate for access to this PDL file. See the element descriptions in Chapter 3.

<VariScript rendering parameters>

Define a value for any VariScript rendering parameter which will be applied to all of the templates in this specific PDL file. These elements include PSAnchor, PSLength, PSN Planes, PSOrientation, PSResX, PSResY, PSRotation, PSScale, PSScaleX, PSScaleY, PSTranslateX, PSTranslate Y, and PSWidth. See the element descriptions in Chapter 4.

Applying a VariScript rendering parameter here will override the definition of the same parameter in the configuration file and in the [JobSetup] group. This definition, in turn, can be overridden by the definition of the same parameter in the [Template Tag] group.

Templates =

See the Templates element description.

SubTemplate =

See the Sub Template element description.

SubTemplate Area =

See the Sub Template Area element description.

Example

[Cover Form]

File pate = forms/cover.ps

Templates = tempA, tempB, tempC, tempD

TEMPLATES

An element that provides a list of descriptive tags which you create to represent the names of the templates in a PDL file.

Syntax

Templates = *Template Tag A, Template Tag B, Template Tag Z*

Remarks

Templates is an element within the user-defined [*PDL File Tag*] group.

NOTE: Each descriptive template tag that you create is for use within the Job Ticket only and is not used outside of that context.

The template tags must be unique within the print job and must appear in the order in which the templates are defined in the PDL file(s). A template that requires no further definition can be represented by a blank tag.

Each tag that appears as a parameter of Templates may become a user-defined group name listed elsewhere in the Job Ticket. The Templates statement simply lets VariScript know that it is to look for user-defined groups and then provides the names (tags) of these groups.

Therefore, in theory, a PDL file with ten pages could have up to ten template tags listed within the Templates element. If any templates need not be named, the order of templates can be preserved by inserting blank template tags into the list.

Explanation***Template Tag A***

Create a descriptive name for the first template in this PDL file. This tag is for use within the Job Ticket.

Template Tag B

Create a descriptive name for the next template in this PDL file. This tag is for use within the Job Ticket.

And so on.

Example

`Templates=tempA, tempB, ,tempD`

SUBTEMPLATE

An element that identifies the templates within the specified PDL file as being subtemplates.

Syntax

`SubTemplate = {True | False}`

Remarks

Optional.

A subtemplate is a template or PDL page that is used as an object to be inserted on another page.

A default value for SubTemplate is False.

Explanation	{True False} If all (or most) of the templates within the specified PDL file are to be identified as subtemplates, type True .
	If all (or most) of the templates within the specified PDL file are NOT subtemplates, type False .
Example	SubTemplate = True

SUBTEMPLATE AREA

An element that assigns a subtemplate name and describes the portion of a template to be used as a subtemplate.

Syntax	SubTemplate Area = Name "<SubName>" X <Units> <Unit Type> \
	Y <Units x Unit Type> Width <Units> <Unit Type> \ Height <Unit> <Unit Type>

See Also	SubTemplate
-----------------	--------------------

Remarks	Optional if SubTemplate = True. Ignored if SubTemplate = False.
----------------	---

This element may be used to specify what part of a template should be used as a subtemplate and to give that extracted portion a subtemplate name.

When this element is defined at a PDL file level, the system will recognize the same extracted portion of each template in the file as being a subtemplate.

Explanation**Name "<SubName> "**

This value assigns a specific name to the subtemplate.

Type the word **Name** followed by a space. For "**<SubName> "**, type the name to be assigned to this subtemplate. Enclose the subtemplate name in double quotation marks (" ").

NOTE: A subtemplate will be known by the official name of the template, unless it is given another name through the SubTemplate Area element.

The official template name is defined by the method of highest precedence. The template name of lowest precedence is the default system-generated template name, followed by the template name you physically define on the PDL template. Of highest precedence is the template name defined using the NewName element in the [Template Tag] group.

Example: Name "mysubl"

X<Unit> <Unit Type>

This value identifies the subtemplate's offset, the horizontal distance between the left side of the template and the area to be extracted.

Type the character **X** followed by a space. For **<Units>** , type the horizontal distance from the template's left side to the beginning of the area to be extracted.

An X value of 0 represents a location flush along the left side of the template. Increasing the value of X locates the area to be extracted the defined distance to the right.

This value is expressed in unitized format if the unit type is different from the default unit type defined in the Units element. Possible

<Unit Type> values are:

cm	for centimeters
dm	for decimeters
dots	for dots
ft	for feet
in	for inch (default value)
m	for meters
mils	for mils
mm	for millimeter
nm	for nanometers
pixels	for pixels
pts	for points
pulses	for pulses
yds	for yards

The default value for X is 0.

Y <Units> <Unit Type>

This value identifies the vertical distance between the bottom of the template and the area to be extracted.

Type the character Y followed by a space. For <Units>, type the vertical distance from the bottom of the template to the beginning of the area to be extracted.

A Y value of 0 represents a location flush along the bottom of the template. Increasing the value of Y locates the area to be extracted the defined distance above the bottom of the template.

The Y value is expressed in unitized format if the unit type is different from the default unit type defined in the Units element. Possible <Unit Type> values are listed above.

The default value for Y is 0.

Width <Unit> <Unit Type>

This value identifies the width of the portion to be extracted (starting at the X, Y coordinates).

Type the word **Width** followed by a space. For <Units>, type the width of area to be extracted from the template. This measurement is the X (horizontal) dimension of the area to be extracted.

This default value for Width is equal to the width of the template.

Heights <Units> <Unit Type>

The value identifies the height (length) of the portion to be extracted (starting at the X, Y coordinates).

Type the word **Height** followed by a space. For <Units>, type the height of the area to be extracted from the template. This measurement is the Y (vertical) dimension of the area to be extracted.

This value is expressed in unitized format if the unit type is different from the default unit type defined in the Units element. Possible <Unit Type> values are listed above.

The default value for Height is equal to the width of the template.

Example

[PageDescriptionLanguageFile]

ps1

ps2

[ps1]

File Path = /mydir/mypsfile.ps

Templates = temp1, temp2

PSScale = .8

[ps2]

File Path = /mydir/mysubtmp.ps

SubTemplate = True

Templates = temp3, temp4, temp5, temp6

SubTemplate Area - X 1in Y 2in Width 3in Height 4in

This example depicts two PDL files.

File ps1 is a "normal" PDL file containing two "normal" templates

File ps2 is identified as a subtemplate file (SubTemplate = True). The same area (SubTemplateArea) on each template is ps2 will be defined as a subtemplate. This area represents a 3-inch wide by 4-inch long portion starting from 1 inch to the right and 2 inches above the lower left corner of the template. Since no Name <SubName> value is given, each subtemplate will take the official name of its template.

[TEMPLATE TAG]

A user-defined tag name for a group that provides information about a template and corresponds to a descriptive tag that you create within the Templates element.

Syntax

[*Template Tag*]

< VariScript rendering parameters >

New Name = < Template New Name >

Sub Template =

Sub Template Area =

See Also

Templates

Remarks

Optional.

A separate [*Template Tag*] group is required for each descriptive tag listed in the Templates element which you will identify as a subtemplate or assign a new template name or a template-level marketing parameter.

Explanation

[*Template Tag*]

Take the descriptive tag within the TemplateNames element and write it here as a group name within brackets [].

< VariScript rendering parameters >

Define a value for any VariScript rendering parameter which will be applied to this specific template. These elements include PSAnchor, PSLength, PSNPlanes, PSOrientation, PSResX, PSResY, PSRotation, PSScale, PSScaleX; PSScale Y, PSTranslater X,

PSTranslate Y, and PSWidth. See the element descriptions in Chapter 4.

A rendering parameter applied within this group has the highest precedence. It will override the definition of the same parameter in the configuration file, in the [JobSetup] group, and in the [PDLFileTag] group.

newName =

See the newName element description

SubTemplate =

See the SubTemplate element description.

SubTemplate Area =

See the SubTemplate Area element description.

Example

[TempA]

 newName = lotto_tikt

 PSScale = 0.8125

 SubTemplate = True

NEWNAME

An element that defines a new name for the specified template. This new name overrides the system-generated default template name or the template name that you may have placed directly on this template within the PDL file.

Syntax

newName = <TemplateName>

Remarks **Optional.**

The template receives its name in one of three ways:

The system automatically generates a default template name.
(lowest precedence)

You can physically name the template when you are working
in your design application (for example, QuarkXPress) before
you output to a PDL file.

You can define a new template name in the Job Ticket within
the NewName element. (highest precedence)

VariScript will recognize only one template name for each template.
Therefore, a template name assigned by using a method of higher
precedence will overwrite a name of lower precedence. For example,
a name defined directly on the template will overwrite the system-
generated default, and, in turn, be overwritten by a name defined
using the NewName element.

The NewName element does NOT have a default value.

Explanation **< TemplateNewName >**
Specify the template's new name.

Example **NewName = lotto_tikt**

SUBTEMPLATE

An element that identifies this templates as being a subtemplate.

Syntax SubTemplate = {True | False}

Remarks Optional.

A subtemplate is a template or PDL page that is used as an object to be inserted on another page.

The default value for SubTemplate is False.

Explanation {True | False}

If this template is to be identified as a subtemplate, type True.

If this template is NOT a subtemplate, type False.

Example SubTemplate = True

While the forms of apparatus and procedure herein described constitute preferred embodiments of the invention, it is to be understood that the invention is not limited to such precise embodiments, and that variations can be made therein without departing from the scope of the invention.

What is claimed is:

1. A method for generating a plurality of bitmaps comprising the steps of:
 - (a) generating a template page description language specification, the template specification including template data and associated graphic attributes defining how the template data is to appear on a printed page, the template specification including at least one variable data identifier;
 - (b) generating a plurality of sub-template page description language specifications, each sub-template specification including sub-template data and associated graphic attributes defining how the sub-template data is to appear on a portion of a printed page;
 - (c) interpreting the template specification so as to generate a plurality of template rendering commands, and during the interpreting step, identifying the variable data identifier;
 - (d) saving the plurality of template rendering commands into memory;
 - (e) associating the variable data identifier with the plurality sub-template specifications;
 - (f) accessing a first sub-template specification from the plurality of sub-template specifications;
 - (g) processing the first sub-template specification so as to generate a plurality of first sub-template rendering commands;
 - (h) accessing a copy of the plurality of template rendering commands from memory;
 - (i) merging the copy of the plurality of template rendering commands with the plurality of first sub-template rendering commands so as to provide a first merged plurality of rendering commands;
 - (j) generating a first merged bitmap from the first merged plurality of rendering commands;
 - (k) accessing a next sub-template specification from the plurality of sub-template specifications;
 - (l) processing the next sub-template specification so as to generate a plurality of first sub-template rendering commands;

(m) accessing a copy of the plurality of template rendering commands from memory;

(n) merging the copy of the plurality of template rendering commands with the plurality of next sub-template rendering commands so as to provide a next merged plurality of rendering commands; and

(o) generating a next merged bitmap from the next merged plurality of rendering commands.

2. The method of claim 1, wherein steps (k) through (o) are repeated for the remaining plurality of sub-template specifications.

3. The method of claim 1, wherein:

the variable data identifier includes a associated graphic attribute;

prior to the merging step (i), the graphic attribute associated with the variable data identifier is saved to memory;

the merging step (i) includes the steps of accessing the graphic attribute associated with the variable data identifier from memory and applying it to the plurality of first sub-template rendering commands so that a bitmap generated by the plurality of first sub-template rendering commands will include the graphic attribute associated with the variable data identifier; and

the merging step (p) includes the steps of accessing the graphic attribute associated with the variable data identifier from memory and applying it to the plurality of next sub-template rendering commands so that a bitmap generated by the plurality of next sub-template rendering commands will include the graphic attribute associated with the variable data identifier.

4. The method of claim 1, wherein the variable data identifier includes an associated graphic attribute that is discarded during the processing step.

5. The method of claim 1, wherein:

wherein the associating step includes the step of referring to a file containing a name corresponding to at least one of the locations and identities of sub-template specifications; and

the file includes a definition of a new graphic attribute to be applied to the sub-template specifications associated with the name.

6. The method of claim 5, wherein the new graphic attribute includes the location of sub-template bitmaps in merged bitmaps.

7. The method of claim 5, wherein the new graphic attribute includes the size of sub-template bitmaps in merged bitmaps.

8. A method for generating a plurality of bitmaps comprising the steps of:
(a) generating a template page description language specification, the template specification including template data and associated graphic attributes defining how the template data is to appear on a printed page, the template specification including at least one variable data identifier;

(b) generating a plurality of sub-template page description language specifications, each sub-template specification including sub-template data and associated graphic attributes defining how the sub-template data is to appear on a portion of a printed page;

(c) processing the template specification so as to generate a template bitmap, and during the processing step, identifying the variable data identifier;

(d) saving the template bitmap into memory;

(e) associating the variable data identifier with the plurality of sub-template specifications;

(f) accessing a first sub-template specification from the plurality of sub-template specifications;

(g) processing the first sub-template specification so as to generate a first sub-template bitmap;

(h) accessing a copy of the template bitmap from memory;

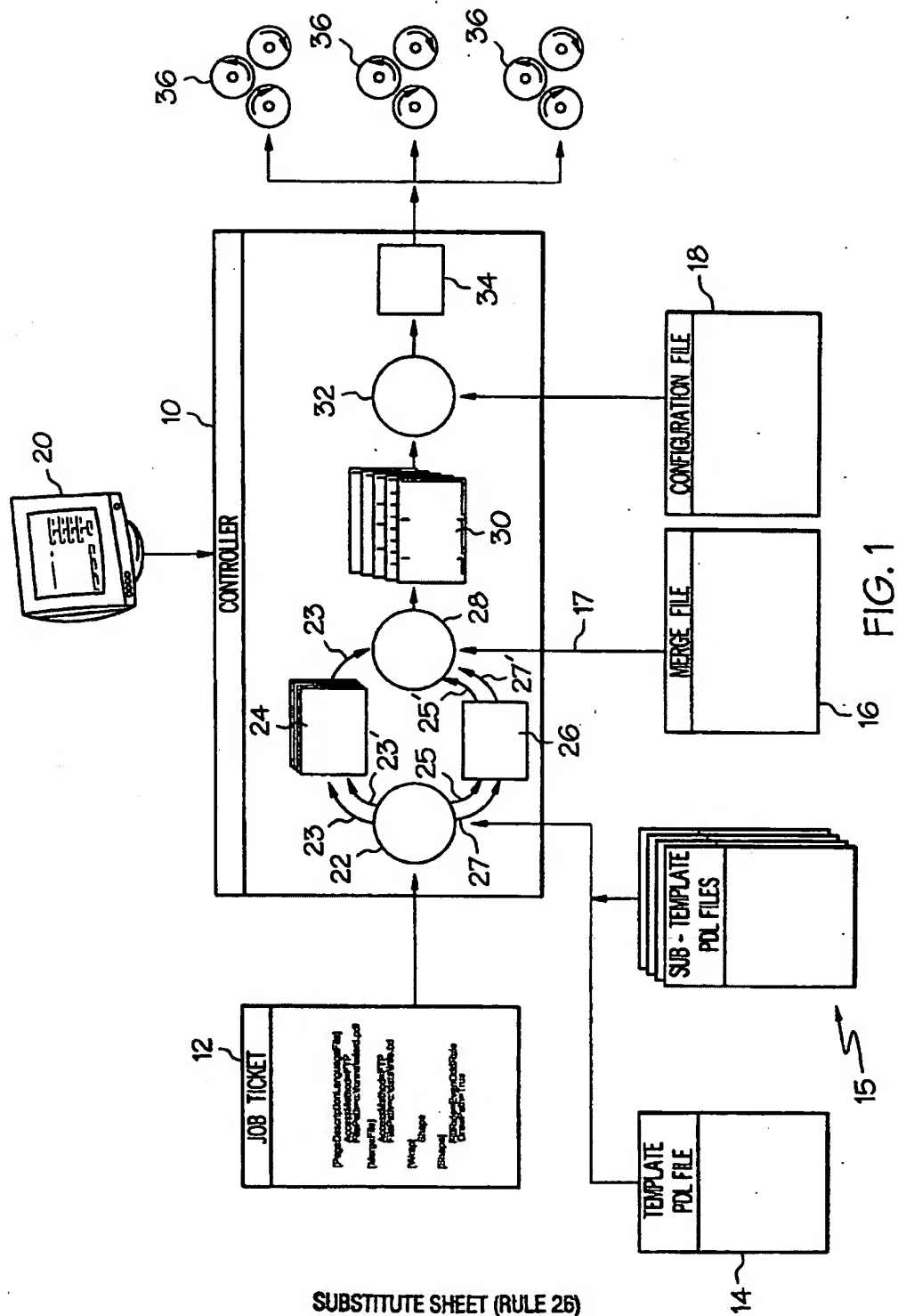
(i) merging the copy of the template bitmap with the first sub-template bitmap so as to provide a first merged bitmap;

(j) accessing a next sub-template specification from the plurality of sub-template specifications;

(k) processing the next sub-template specification so as to generate a next sub-template bitmap;

(l) accessing a copy of the template bitmap from memory; and

(m) merging the copy of the template bitmap with the next sub-template bitmap so as to provide a next merged bitmap.



2 / 8

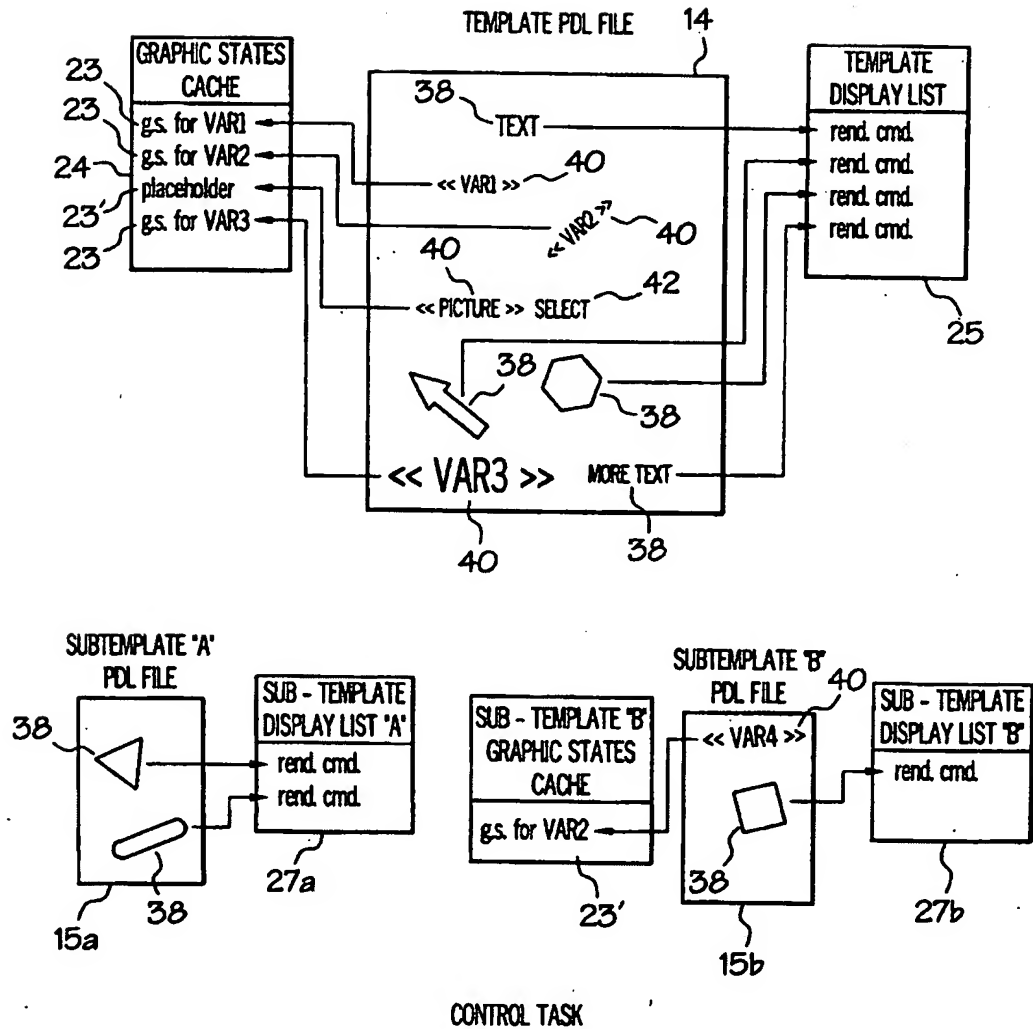
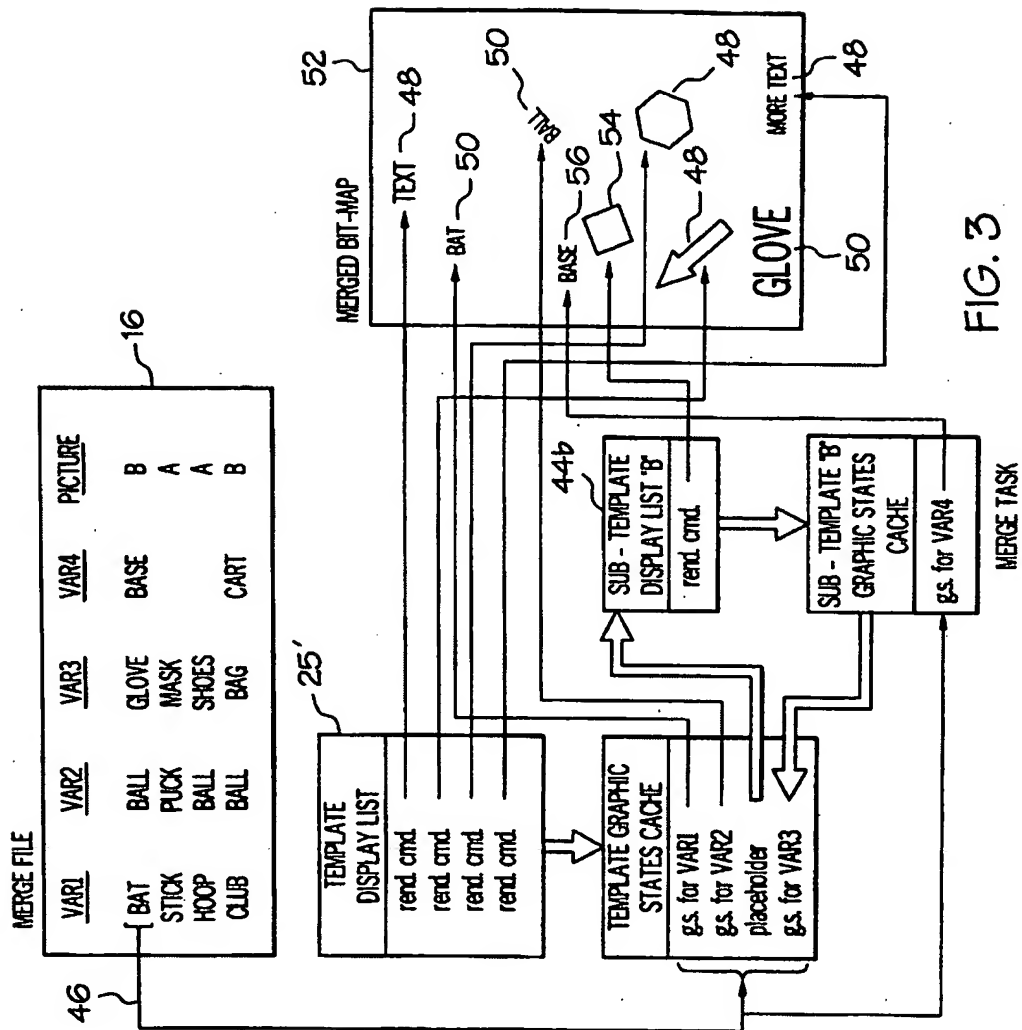


FIG. 2



4 / 8

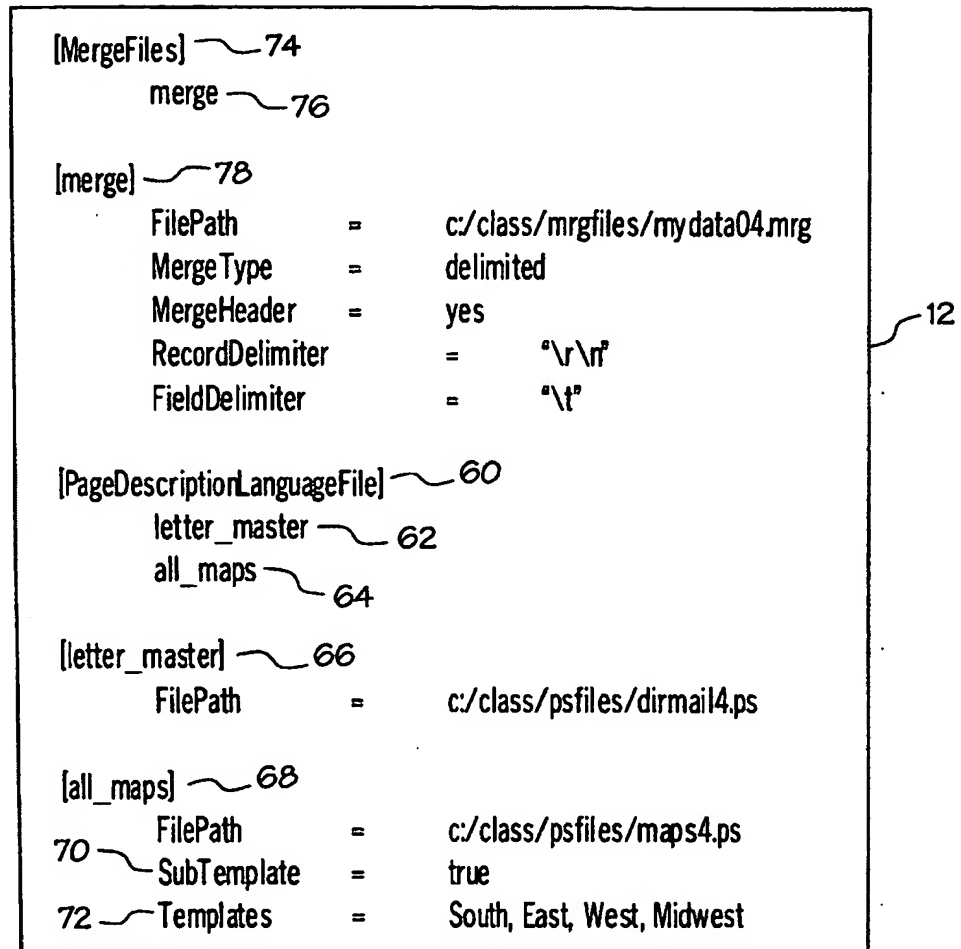


FIG. 4

-16

86

FIG. 5

6/8

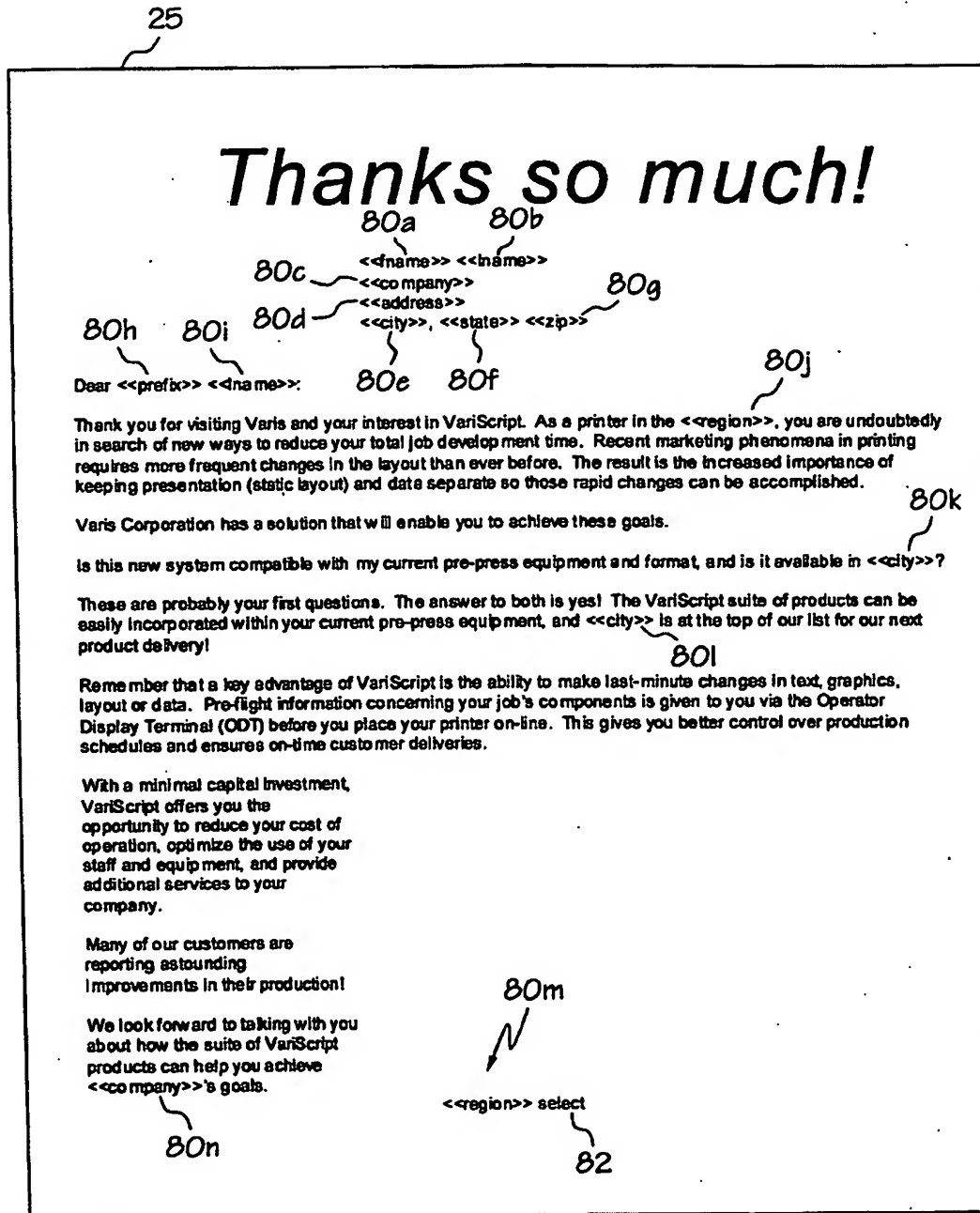


FIG. 6

SUBSTITUTE SHEET (RULE 26)

718

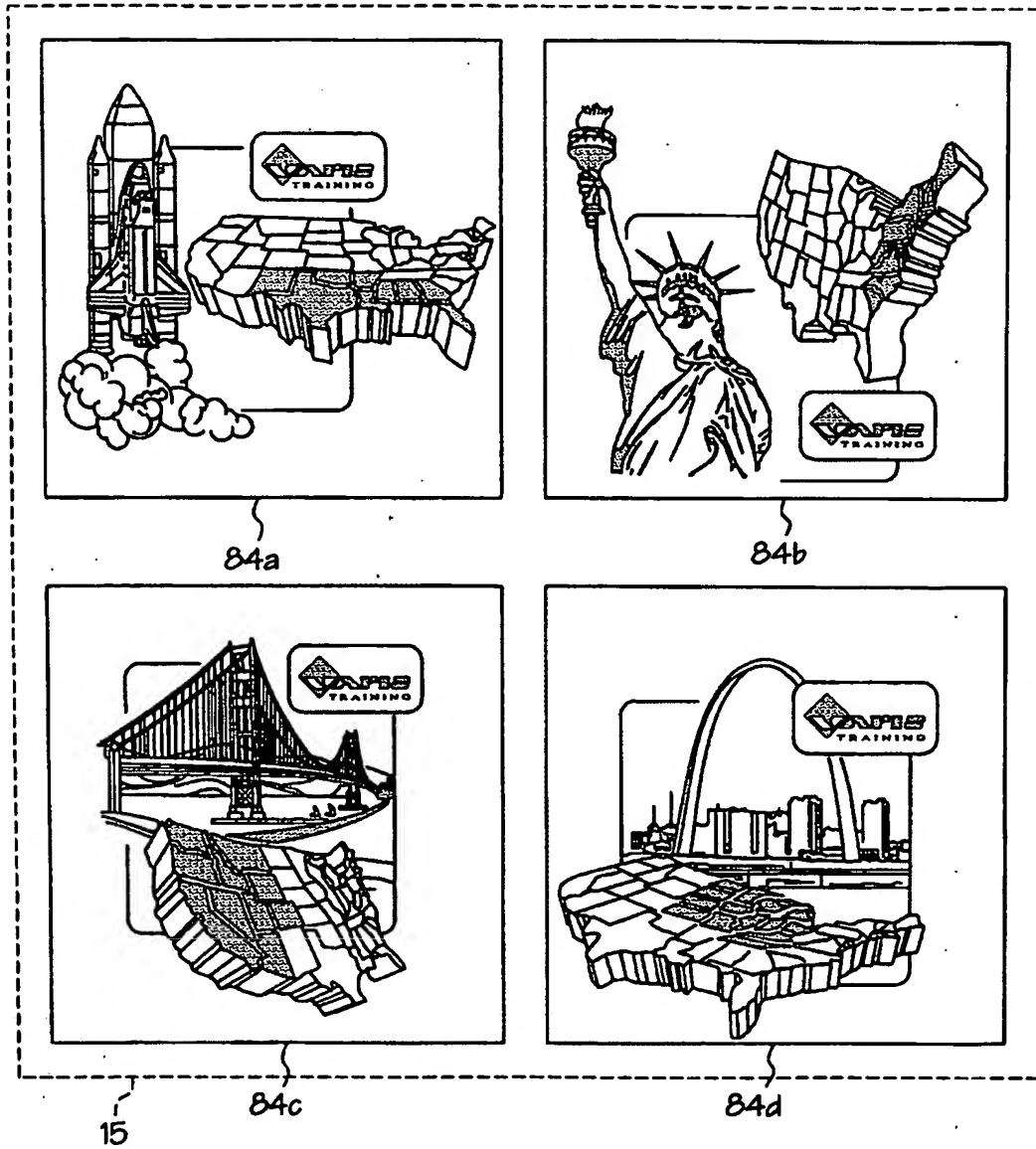


FIG. 7

8/8

Thanks so much!

88a 88b
Shannon Janszen
88c Golf World
88d 88626 Bannister Street
Venice, FL 34293-9183
88h 88i 88e 88f 88g 88j

Dear Ms. Janszen:

Thank you for visiting Varis and your interest in VariScript. As a printer in the South, you are undoubtedly in search of new ways to reduce your total job development time. Recent marketing phenomena in printing requires more frequent changes in the layout than ever before. The result is the increased importance of keeping presentation (static layout) and data separate so those rapid changes can be accomplished.

Varis Corporation has a solution that will enable you to achieve these goals.

Is this new system compatible with my current pre-press equipment and format, and is it available in Venice?

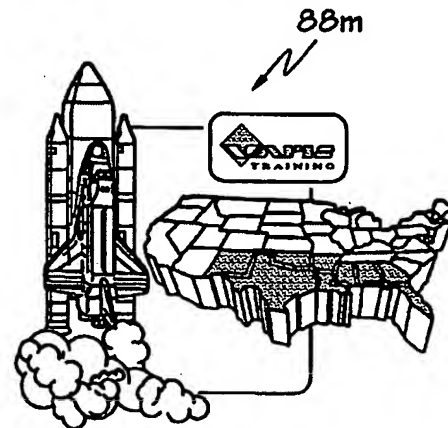
These are probably your first questions. The answer to both is yes! The VariScript suite of products can be easily incorporated within your current pre-press equipment, and Venice is at the top of our list for our next product delivery!

Remember that a key advantage of VariScript is the ability to make last-minute changes in text, graphics, layout, or data. Pre-flight information concerning your job's components is given to you via the Operator Display Terminal (ODT) before you place your printer on-line. This gives you better control over production schedules and ensures on-time customer deliveries.

With a minimal capital investment, VariScript offers you the opportunity to reduce your cost of operation, optimize the use of your staff and equipment, and provide additional services to your company.

Many of our customers are reporting astounding improvements in their production!

We look forward to talking with you about how the suite of VariScript products can help you achieve Golf World's goals.



88n

FIG. 8

SUBSTITUTE SHEET (RULE 26)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/09646

A. CLASSIFICATION OF SUBJECT MATTER IPC(7) : G06F 15/00 US CL : 358/1.18 According to International Patent Classification (IPC) or to both national classification and IPC				
B. FIELDS SEARCHED Minimum documentation searched (classification system followed by classification symbols) U.S. : Please See Extra Sheet. Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)				
C. DOCUMENTS CONSIDERED TO BE RELEVANT				
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.		
A	US 6,016,380 A (NORTON) 18 January 2000, whole document.	1-8		
A,P	US 5,937,153 A (GAUTHIER) 10 August 1999, whole document.	1-8		
A	US 5,852,673 A (YOUNG) 22 December 1998, whole document.	1-8		
A	US 5,796,411 A (CYMAN et al.) 18 August 1998, whole document.	1-8		
A	US 5,740,338 (GAUTHIER et al.) 14 April 1998, whole document.	1-8		
A	US 5,729,665 A (GAUTHIER) 17 March 1998, whole document.	1-8		
<input checked="" type="checkbox"/> Further documents are listed in the continuation of Box C. <input type="checkbox"/> See patent family annex.				
<table border="0"> <tr> <td> * Special categories of cited documents: *A* document defining the general state of the art which is not considered to be of particular relevance *B* earlier document published on or after the international filing date *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) *O* document referring to an oral disclosure, use, exhibition or other means *P* document published prior to the international filing date but later than the priority date claimed </td> <td> *T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art *A* document member of the same patent family </td> </tr> </table>			* Special categories of cited documents: *A* document defining the general state of the art which is not considered to be of particular relevance *B* earlier document published on or after the international filing date *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) *O* document referring to an oral disclosure, use, exhibition or other means *P* document published prior to the international filing date but later than the priority date claimed	*T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art *A* document member of the same patent family
* Special categories of cited documents: *A* document defining the general state of the art which is not considered to be of particular relevance *B* earlier document published on or after the international filing date *L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified) *O* document referring to an oral disclosure, use, exhibition or other means *P* document published prior to the international filing date but later than the priority date claimed	*T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention *X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone *Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art *A* document member of the same patent family			
Date of the actual completion of the international search 02 JULY 2000		Date of mailing of the international search report 27 JUL 2000		
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230		Authorized officer GABRIEL I. <i>James R. Matthews</i> Telephone No. (703) 305-3800		

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/09646

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,506,697 A (LI et al.) 09 April 1996, whole document.	1-8
A	US 5,459,826 A (ARCHIBALD) 17 October 1995, whole document.	1-8
A	US 5,291,243 A (HECKMAN et al.) 01 March 1994, whole document.	1-8

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/09646

B. FIELDS SEARCHED

Minimum documentation searched

Classification System: U.S.

358/1.1, 1.9, 1.15, 1.17, 1.18, 1.19, 448, 467, 470, 518, 520, 538; 382/171, 180, 209, 215, 217, 277, 302; 345/113, 433, 435, 436, 438, 468, 522.